Office Action Summary	Application No.	Applicant(s)	
	09/765,916	CANUT ET AL.	
	Examiner	Art Unit	
	INSUN KANG	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 July 2011*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

5)☒ Claim(s) *1-31* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-31* is/are rejected.

8)☐ Claim(s) _____ is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

12)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☒ Interview Summary (PTO-413) Paper No(s)/Mail Date. *20111218*.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the RCE amendment filed on 7/26/2011.

2. Claims 1-31 are pending in the application.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-3, 8-16 and 21-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Pieper et al (US 2003/0005419) in view of Cain et al ("Portable Software Library Optimization," 2/1998) hereinafter referred to as "Cain."

Regarding claim 1:

Pieper et al. disclose: a method of optimizing a software program for a target processor to meet performance objectives, where the software program is coded in a high-level Language (par. 0019; par. 0020), the method comprising the steps of: (a) optimizing the software program such that, determining a first performance profile for the first optimized form of the software program, and comparing the first performance profile with the performance objectives (par. 0020; 0030).

Pieper et al. do not explicitly disclose that a resulting first optimized form of the software program is completely independent of the target processor and is at least

partially coded in the high-level language. However, Cain teaches that such a portable optimized high-level source code was known in the art of software development and optimization, at the time applicant's invention was made, to provide portability to different platforms (i.e. section1, page 1, second paragraph). It would have been obvious for one having ordinary skill in the art of computer software development and optimization to modify Pieper's disclosed system to incorporate the teachings in Cain. The modification would be obvious because one having ordinary skill in the art would be motivated to maintain portability of programs that are optimized in high-level target independent code in Pieper.

Pieper et al. in view of Cain further discloses:

(b) based on results of comparing the first performance profile with the performance objectives, if the performance objectives are not met by the first optimized form of the software program, then optimizing the first optimized form of the software program such that a resulting second optimized form of the software program includes at least one portion that is dependent on the target processor and is coded in the high-level language, wherein the at least one portion of the second optimized form of the software program is less than an entirety of the second optimized form (par. 0031, 0020; par. 0045);

Pieper et al. do not explicitly disclose flagging at least one portion to indicate that the at least one portion is dependent on the target processor if the first optimized form of the software program is optimized to create the second optimized form of the software program. However, Cain teaches that using flags was known in the art of

software development and optimization, at the time applicant's invention was made, to mark or identify some portions or whole code as an event of some type or having a special purpose or capability ("#include directive is used to retrieve the desired system-specific API," page 7). It would have been obvious for one having ordinary skill in the art of computer software development and optimization to modify Pieper's disclosed system to incorporate a target dependent code in a high level language using a flag for the target dependent code. The modification would be obvious because one having ordinary skill in the art would be motivated to identify the target specific code in a source code for more aggressive optimization in a high level language and portability (page 6-7) as taught by Cain.

Pieper further discloses wherein the acts of optimizing are performed such that the first and second optimized forms of the software program are progressively more dependent on the target processor (i.e. 0030;0031).

Regarding claim 2:

The rejection of claim 1 is incorporated, and further, Pieper et al. disclose: (bl) determining a second performance profile for the second optimized form of the software program, and comparing the second performance profile with the performance objectives (par. 0032; 0044) as claimed.

Regarding claim 3:

The rejection of claim 2 is incorporated, and further, Pieper et al. disclose:

-optimizing the second optimized form of the software program such that a resulting third optimized form of the software program is at least partially dependent on the target processor and includes portions coded in a low-level language of the target processor (par. 0031) as claimed.

Regarding claim 9:

Pieper et al. further disclose the act of implementing reference code comprises code profiling (par. 0031, 0042 ; 0046 ; 0048 ; 0049 ; 0052) as claimed.

Regarding claim 8, this claim is another version of the claimed method discussed in claim 9, wherein all claim limitations also have been addressed and/or covered in cited areas as set forth the above.

Regarding claim 10:

The rejection of claim 1 is incorporated, and further, Pieper et al. disclose :

-the act of optimization predicted to improve resulting assembly code ("In generating the code, generator modifies the code such that code reflects scheduling and other low-level optimizations of the code, which are dependent on the target processor architecture," 0031; 0032; 0009).

Regarding claim 11:

The rejection of claim 1 is incorporated, and further, Pieper et al. disclose the act of tuning low-level functions (0031) as claimed.

Regarding claim 12:

The rejection of claim 1 is incorporated, and further, Pieper et al. disclose the act of manual assembly optimization. Hand-coded assembly for optimized performance is

necessary for performance critical routines such as graphics or math library routines as they often must access low-level machine instructions for optimal execution performance. Therefore, accordingly, Pieper et al. anticipate this claim. See also 0009 and 0018.

Regarding claim 13:

The rejection of claim 1 is incorporated, and further, Pieper et al. the act of feature tuning (0031; 0032).

Per claim 27:

Pieper et al. further discloses: wherein the second optimized form of the software program includes the at least one portion that is dependent on the target processor and another portion that is independent of the target processor (par. 0031, 0020; par. 0045).

Per claim 28:

Pieper et al. further discloses: wherein the act of optimizing the first optimized form
of the software program uses a subset of the first optimized form (par. 0031, 0020; par. 0045).

Per claims 14-16, 21-26, 29, and 31, they are the computer-readable medium versions of claims 1-3, 8-13, 27, and 28, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 1-3, 8-13, 27, and 28 above.

Per claim 31, it is the system version of claim 1, respectively, and is rejected for the same reasons set forth in connection with the rejection of claim 1 above.

5. Claims 4-7 and 17-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Pieper et al (US 2003/0005419) in view of Cain et al ("Portable Software Library Optimization," 2/1998) hereinafter referred to as "Cain" and further in view of Kum et al. (0-7803-5041-3/99, IEEE).

Regarding claim 4:

The rejection of claim 1 is incorporated, and further, Pieper et al. and Cain do not explicitly teach a floating-point implementation. However, Kum et al. disclose deriving a floating point implementation (pg 2163, introduction, par. 3, "the ranges of floating point variables are estimated by the simulation of the range estimation program that is automatically generated from the original floating-point version," see also Figure 1) for the purpose of automatic scaling of all numbers so that the numbers use the full word length available and for the purpose of reducing the risk of overflow. Therefore, it would have been obvious to a person of ordinary skill in the art to incorporate the teachings of Kum et al. to the system of Pieper et al and Cain. The modification would be obvious to include the floating-point implementation because of the automatic scaling of each number to use the full word length of the mantissa so that accurate representation of numbers can be obtained while minimizing the risk of overflow and quantization errors (pg 2163, introduction, par. 3).

Regarding claim 5:

The rejection of claim 1 is incorporated, and further, Pieper et al. and Cain do not explicitly teach a fixed point implementation. However, Kum et al. disclose the

method of claim 1 in which step (a) comprises the act of deriving a fixed point implementation so that "assembly coding and manual scaling can be avoided and the translated C programs are executed very efficiently" in fixed-point DSPs (pg 2163, introduction, lines 1-15). Therefore, it would have been obvious to a person of ordinary skill in the art to incorporate the teachings of Kum et al. to the system of Pieper et al and Cain. The modification would be obvious to include the fixed-point implementation so that round-off errors can be prevented and target dependent scaling shift can be minimized while obtaining fast real-time processing with less power and memory usage (pg 2163, introduction, lines 1-15).

Regarding claim 6:

The rejection of claim 5 is incorporated, and further, Pieper et al. and Cain do not explicitly teach the act of processing qualification. However, Kum et al. further disclose the act of processing qualification (Introduction, par.3; simulation-based integer word-length determination, pg 2165, shift reduction, par. 10; pg 2163, par. 6; pg 2166, Concluding remarks) so that cost effective and high quality fast real-time processing with less power and memory usage can be obtained while reducing quantization noise (Introduction, par.3; simulation-based integer word-length determination, pg 2165, shift reduction, par. 10; pg 2163, par. 6; pg 2166, Concluding remarks). Therefore, it would have been obvious to a person of ordinary skill in the art to incorporate the teachings of Kum et al. to the system of Pieper et al and Cain. The modification would be obvious to include the act of processing qualification for the purpose of high quality processing with minimized quantization noise.

Regarding claim 7:

The rejection of claim 5 is incorporated, and further,, Pieper et al. and Cain do not explicitly teach the act of implementation sizing. However, Kum et al. further disclose the act of implementation sizing (abstract; Introduction, pg 2163, par.3; pg 2163, simulation-based integer word-length determination) by program-profiling results (pg 2164-2165, Sift reduction) so that estimation of code size for the target can be obtained and the risk of overflow can be prevented. Therefore, it would have been obvious to a person having ordinary skill in the art to incorporate the teachings of Kum et al. to the system of Pieper et al and Cain. The modification would be obvious to include the act of implementation sizing for the purpose of code size estimation so that the risk of overflow can be prevented (pg 2164-2165, Sift reduction).

Per claims 17-20, they are the computer-readable medium versions of claims 4-7, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 4-7 above.

Response to Arguments

6. Applicant's arguments filed on 7/26/2011 have been fully considered but they are not persuasive.

The applicant states that in Pieper, the code 60 is always generated, and it is not generated based upon whether performance objective is met by a first optimized form that is completely independent of target processor.

In response, the optimized code produced by the optimization processes 58 based on execution profile data 78 generated by execution analysis process 76 is high-level

optimization that is machine-independent (0020; 0030). The high-level optimization is applied to the intermediate code. The optimization process uses the profile data to generate a new more efficient version of form 60 only if the final code does not exhibit optimal execution performance (0032).

The applicant states that the execution profile data in Pieper is generated based on code 64 that is completely dependent on a target processor and is not based on whether objective profile is met by a first optimized form that is completely independent on target processor. Pieper actually teaches away from generating code 60 based on performance profile from a code that is completely independent on target processor.

In response, in the instant specification, there is no description of a profile. It merely states performing the code profiling operations at page 9. Therefore, the profiling operation is considered to be a conventional profiling that gathers execution information of a code as in Pieper. Also, it is noted that the optimization process 58 is applied to the intermediate code 56 that is machine-independent, therefore, it is reasonable to interpret that the execution profile data 78 is based on whether the optimal execution performance is exhibited by the optimization process at 58 that is machine independent.

The applicant states that Pieper and Cain do not disclose the second optimized form (code 64) is conditioned whether performance objective is met by the first optimized form 60 that is less target dependent than the second optimized form. The modification to combine Pieper and Cain would not have been obvious because Pieper specifically requires that a target machine model 80 be input into the optimization process 58.

In response, the target machine model 80 is not in a machine dependent form, rather, it is a high-level **model** which is supplied to the high-level optimization to the intermediate code (see figure 2). The target machine model is supplied to ensure adequate prefetch analysis (i.e. 0062; 0063). As the result from the optimization process 58 is not machine-dependent and Cain clearly teaches a portable optimized high-level source code (i.e. section1, page 1, second paragraph), the motivation to combine Pieper and Cain would have been obvious to maintain portability of programs that are optimized in high-level target independent code in Pieper. In the instant application, the target dependent portion is merely encapsulated by a compiler flag inserted into a source code which is a widely used technique in the prior art. Specifically Cain discloses such compiler directives which are inserted to the source code to encapsulate a system specific code for portability (see page 7).

The applicant states that Cain does not disclose flagging. Cain does not disclose or suggest that any act of flagging is conditioned upon whether the first optimized form of the software program is optimized to create the second optimized form of the software program.

In response, the instant specification merely recites flagging to indicate a target-specific code and the use of #if-define or other such conditional compiling flags integrated into the code so that it is possible to recompile the same application for all the targets to be addressed. Even though Pieper does not address a portability issue and is silent of incorporating target specific portion into the source code by using a compiler directive for more aggressive optimization in a high level language, Cain clearly discloses the #ifdef directives that are for “compile-time conditional code compilation”

and "used to retrieve the desired system-specific API," (page 7)." The use of #ifdef flag for target specific code in Cain is also to achieve software portability by encapsulating the system-specific sections in Cain's portable software optimization (i.e. pages 3, page 7). Therefore, the modification to use the #ifdef flag to encapsulate the target-dependent portion in a source code in Pieper would be obvious to a person having ordinary skill in the art to identify the target specific portion in a high level language for portability as taught by Cain.

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to INSUN KANG whose telephone number is (571)272-3724. The examiner can normally be reached on M-R 7:30-6 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis A. Bullock, Jr. can be reached on 571-272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO

Customer Service Representative or access to the automated information system, call

800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Insun Kang/
Primary Examiner, Art Unit 2193